CHAPTER

12

DEPLOYING AND MANAGING SOFTWARE BY USING GROUP POLICY

After completing this chapter, you will be able to:

- Understand IntelliMirror components and change and configuration management concepts
- Understand the phases of software management
- Create and configure Windows Installer packages
- Manage software deployment through the Software Installation MMC snap-in
- Patch software using the Windows Installer Package Editor

To this point, we have discussed a couple of the key components of change and configuration management: user data management (folder redirection) and user environment management. In this chapter, we move into the final phase: software installation and maintenance.

Windows 2000, unlike its NT predecessors, provides the ability to manage software throughout an organization by using a centralized Group Policy–based management system. This management system reduces total cost of ownership (TCO) for an organization, because it allows software to be efficiently managed remotely, without you having to go to every separate computer in order to make changes. Remote management reduces the number of support personnel that must be kept on staff, which translates into cost savings for a company.

In this chapter, we will examine the components of a software installation and management strategy, from basic concepts through best practices. In between, you'll learn how to build Windows Installer packages and how to install, patch, and remove software through Group Policy. Without further ado, let's begin!

IntelliMirror Components and Change and Configuration Management Concepts

The last several chapters have focused extensively on topics that make up the area known as **change and configuration management**, which is a collection of ideas and strategies for reducing TCO and increasing return on investment (ROI). Although these may sound like topics better suited for the bean counters in the company, it has become increasingly important for the IT professional to be business savvy and to be able to evaluate IT needs from business perspectives. Change and configuration management includes tasks such as documentation and (as the name implies) managing changes and configurations within an organization. That may sound a little redundant, so let's examine this concept further.

From an IT perspective, change within a company usually refers to new hardware and software being deployed; configuration generally refers to the standard configuration of hardware and software. Managing change involves strategic planning for prepared change (rolling out new equipment) and responsiveness for unplanned change (for example, the server network interface card [NIC] goes bad and is replaced by a newer model NIC). Managing configurations typically involves maintenance and support—for example, responding to a help desk call when a user inadvertently deletes several key system files. Why is change and configuration management so important? The answer is easy to find: Just look at the bottom line of the general ledger.

As we said previously, the goal of change and configuration management is to reduce TCO and increase ROI. An example of a real-world situation dealing with TCO issues can help illustrate why these objectives are so important.

Hard versus Soft Costs

Many IT professionals never consider the **soft costs** of an IT infrastructure, but rather look only at the **hard costs** (the initial cost of the equipment). For example, say you buy 500 new systems at $2,500 each. The hard cost is $1.25 million for those systems. But also consider that users are being paid to utilize the systems, and a staff of IT professionals is being paid to support them. Say one of your systems crashes, for whatever reason, causing an IT staff member to have to go to the user's office and fix the computer. A cost is associated with this event, because the user's salary is paid during the downtime when he cannot work, and the IT staff member's salary is paid even though she is unavailable for work on other projects while spending time fixing the computer. Depending on the type and number of problems a company has on any given day, the number of IT staff required to support the computers grows. The costs of support and lost productivity increase the TCO of the computer to a number far higher than the initial $2,500 spent on each computer. These are the soft costs associated with information technologies, and it is often extremely difficult to place an exact dollar figure on these costs.

Many businesses now demand that IT departments function more as business units and less as cost centers. Therefore, although IT doesn't generate direct revenue like a sales staff, it can dramatically reduce expenditures and increase productivity through change and configuration management.

Windows 2000 supports a number of new change and configuration management features that enable a company to realize tremendous savings in TCO. The entire next chapter is devoted to one of the features, Remote Installation Services (RIS). The other tool is a collection of features collectively referred to as **IntelliMirror**.

IntelliMirror

IntelliMirror is a feature of Windows 2000 that seeks to increase the availability of Windows 2000–based computers while decreasing TCO. In the past, you may have heard of IntelliMirror as Zero Administration Windows (ZAW), which was Microsoft's original initiative to create a managed desktop environment under Windows. IntelliMirror relies on a combination of Active Directory and Group Policy in order to be deployed, and it consists of three key features. We have already discussed two of these features at great length, and the third feature is the focus of this chapter.

The features of IntelliMirror are:

- *Data management*—Data management is implemented in Windows 2000, as we have seen, through folder redirection. When folder redirection and offline folders are used, user data can be synchronized between a server copy and a local copy, ensuring that data files are accessible no matter where users are and what computer they log on from.
- *Desktop settings management*—Desktop settings can be stored in profiles that roam with a user so that they are applied whenever a user logs in to a networked computer. Group Policy is used to control what settings should be stored and can control a user's ability to make changes to desktop settings.
- *Software installation and maintenance*—The focus of this chapter, this feature of IntelliMirror allows applications to be published by an administrator for use by defined users, computers, and groups. Windows Installer applications can even be set up to replace corrupt or missing files automatically. Doing so reduces downtime associated with broken applications that would ordinarily require an IT staff member to go to a workstation and manually reinstall the application.

The great advantage of using IntelliMirror technologies is that usually users don't know what is happening behind the scenes. The automation that IntelliMirror provides often avoids problems that would otherwise require intervention from a desktop support technician. In addition, IntelliMirror can provide a consistent computing experience for users no matter what computer or location they log on from, which is something that could not be done in the past.

A Real-World Example

To illustrate the effectiveness of IntelliMirror technologies, here's a real-life example of how employing IntelliMirror dramatically enhanced the end-user experience and decreased TCO for a company.

The company I work for is a fairly standard Microsoft shop, using BackOffice on the back end and a mixture of NT 4 Workstation and Windows 98 on the front end. We had about a dozen outside sales people who would come into the office periodically; usually, at least two or three would be in the office each day. When meetings were held, they took place in the office. Because the sales staff was outside sales, they worked from their homes rather than having cubicles or offices at the corporate location. When they were in the office, however, they needed to be able to check their e-mail from whatever computer they could beg, borrow, or steal for a few minutes.

Our e-mail environment uses Exchange Server and Outlook 2000, and with Outlook one must configure an e-mail profile in order to access the e-mail account. Dealing with e-mail profiles was a constant support issue. Either a salesperson would configure and change the default profile of a computer, causing the regular users of the PC to get an error when they came back to their PC, or the salesperson would not know how to create a profile and would require assistance. Because a particular salesperson did not always use the same computer when in the office, it was difficult to ensure that the salespeople could always access their e-mail, short of going to each computer and creating e-mail profiles for every salesperson. If we had done this, we would have had to train both the sales staff and the regular users on how to manipulate profiles, in order to ensure that the correct profile was selected for a given user. In addition, whenever there was a personnel change, the process would have to be repeated. This would definitely be a high-maintenance process, which would have driven up TCO for our e-mail system.

Our initial solution, before Windows 2000, was to attempt to use the Outlook Web Access (OWA) feature of Exchange Server and Internet Information Server (IIS), which allows users to access their e-mail accounts through Internet Explorer. Because of certain performance and security issues with OWA, however, that solution was abandoned.

After upgrading our network to Windows 2000, we were able to use IntelliMirror to create a situation in which the IT staff did not have to intervene to ensure that whoever logged into a computer got the correct e-mail profile. Sales staff members, regular users, or anyone else with a valid domain account could log on to any computer on the network, and their settings would automatically be applied. The specific IntelliMirror technologies we used to make this happen were:

- *Active Directory*—The cornerstone. Without AD, none of the rest would be possible.
- *Group Policy*—Through Group Policy, we could manage desktop settings and determine what to apply and where.

- *Roaming user profiles*—Through roaming profiles, we were able to ensure that when users logged in to a computer, their settings were applied. This was the key to having e-mail profiles follow their owners.
- *Folder redirection*—In addition to profiles, we redirected user folders to a network share so they would be backed up. As you learned in Chapter 11, folder direction is not applied over slow links; so it was not in effect for the sales staff when they dialed in from home over 56Kbps modems.
- *Offline folders*—In addition to the outside sales staff, some users work both in the office and at home or on the road. These users have laptops, and we were able to use offline folders in conjunction with folder redirection to ensure they have full access to their files regardless of whether they are in the office on the LAN or working offline on an airplane.

As you can see, IntelliMirror technologies increased the availability of resources to a variety of types of users, while reducing the support burden on the IT staff at the same time. Since implementing IntelliMirror, the IT staff has been able to focus on more time-intensive projects that were difficult to complete in the past due to the support volume. Because we are able to better use our staff's time, we have translated the increased productivity into cost savings, which ultimately has led to better bonuses for each of us (the company uses a profit-sharing pool). Both users and IT staff members are happier on a day-to-day basis, as well. The users are happier because they get a consistent computing experience and do not have to call the help desk as often, and the IT staff is happier because they don't have to perform as many desktop support tasks and can spend time on more interesting projects.

Having already covered data management and settings management, we will spend the rest of this chapter discussing the software installation and management aspect of IntelliMirror.

Understanding the Phases of Software Management

This section will present a high-level overview of software management by taking you through the stages—or phases—of the deployment of a new software application. Breaking down software management into phases provides a systematic, repeatable method for upgrading and deploying software in an organization. Your high school science teacher probably talked frequently about systematic methods for scientific experiments, in which the goal is to perform an experiment in a way that can be repeated each time and provide consistent results. That is essentially what we are doing in the IT world: We want to devise procedures that we can follow every time we roll out a new application and that will give us the positive results we are looking for.

The phases of software management that we will discuss here are:

- Preparation phase
- Distribution phase

- Targeting phase
- Pilot phase
- Installation phase

Preparation Phase

The preparation phase of software management consists of the initial information collecting process. This phase includes analyzing your organization's structure to determine its software requirements. In addition, you will need to determine how many licenses you must purchase in order to be legal, and decide how users will access the software. The following are some of the questions you will need to answer during the preparation phase:

- Will the software be loaded on a central server that users will access, or will the software be installed locally on users' computers? The answer to this question will help determine capacity requirements on the servers and workstations, and will dictate how the application is set up for distribution.
- What users will need access to the new application? If not everyone needs access, you must arrange through Active Directory and Group Policy for the required users to have access while ensuring that users who shouldn't have the application do not have access. You might have to create additional Organizational Units (OUs) based on software needs.
- Do you have enough licenses or will you need to purchase additional ones? This is an important question, because you do not want to get caught in a situation where you are short on software licenses. There are plenty of horror stories about companies paying extensive fines for using unlicensed software, and you don't need or want your company to become one of them.
- Will the network infrastructure support the deployment design you are considering? For example, if you have multiple locations connected by wide-area network (WAN) links, you won't be able to run server-based applications from one or more central servers at one location. You will need to deploy these applications to remote servers so that all users who should have access to the application are running it over a local-area network (LAN) connection. If applications will be installed directly to users' hard drives, you will need to plan for local distribution points so that software is not installed over a slow WAN connection.

Assigned or Published?

One other key question to consider is whether the application will be assigned directly to users or published into Active Directory. When an application is assigned, its icons appear in the Start menu or on the desktop of the user's computer according to criteria defined by the administrator. The first time a user attempts to launch an assigned application, the software is

automatically installed without user or administrator intervention. If a user later uninstalls the application through the Add/Remove Programs applet in Control Panel, the software will still be available to the user the next time he or she logs on. If a user attempts to launch a program he or she thought had been uninstalled, the software will simply reinstall itself and open. Because the software has been assigned to the user, the user cannot get rid of it. The advantage to this arrangement is that users cannot break assigned applications; rather, the software is self-healing, without requiring a traditional visit from an IT staff member to repair it through manual reinstallation.

On the other hand, administrators can also choose to simply publish applications into the Active Directory rather than assigning them directly to users. Assigned applications are typically used when users need a particular application to do their job. Published applications are not necessarily required by users to do their jobs, but they are beneficial applications that the administrator wants to make available. A published application shows up in Add/Remove Programs but must be explicitly installed by the user. No icons appear in the Start menu or on the desktop in advance to inform users of the application's availability, and if a user uninstalls the application, it is removed from the computer just as an application would be after a traditional uninstall procedure.

In addition to answering the previous questions, during this preparation phase an administrator prepares the software for distribution. This process includes creating any necessary Windows Installer packages (discussed later in this chapter).

Distribution Phase

Once all of the planning and preparing has been completed, you are ready to move on to the distribution phase. In this phase, you define network locations, called **distribution points** in Windows 2000, which will hold the software you want to deploy. Essentially, the main task of the distribution phase is to get the software to the distribution points from which they can be installed.

Distribution points are simply network locations from which users can install software. Creating a distribution point is a matter of creating network shares with the appropriate permissions, and then copying the installation files for an application to the share. For ease of administration and ease of use, you should create subfolders for each application you are making available.

After creating your distribution points, you must either copy the installation files to the distribution points or (in the case of some applications) perform an administrative installation to the distribution points. Administrative installations usually involve invoking a special switch to the command line, such as **setup /a** in the case of Microsoft Office 97. This type of installation allows you to customize certain options as an administrator when you distribute the installation files to a distribution point.

In general, administrators should have full control permissions over the distribution points, and users should have read permissions.

Software Deployment Options

Windows 2000 can distribute Windows Installer packages through the Software Installation snap-in in Group Policy. The limitation, of course, is that Group Policy is supported only on Windows 2000 clients. In addition, Software Installation can support only the deployment of Windows Installer packages. Because not all applications are Windows Installer compatible (and hence Windows 2000 certified), you will often run into applications that cannot be deployed through these means. If you want to distribute non–Windows Installer applications or to distribute to Windows 9*x* or Windows NT 4 clients, you can use Microsoft Systems Management Server (SMS). The use of SMS is outside the scope of this book, but suffice it to say that it provides a wealth of software deployment features well beyond what Windows 2000 can offer out of the box. In many environments, a combination of Windows 2000 Software Installation and SMS is appropriate.

Targeting Phase

Once the software has been distributed through copying or administrative installation to the various distribution points, you are ready to begin the targeting phase of software management. The emphasis during this phase is on determining the scope of management—that is, to determine the needs of the users.

To this point, you have analyzed to whom you want the software to go, and you have established the distribution points that hold the software to be deployed. Now, in the targeting phase, you use Group Policy to create and/or modify Group Policy Objects (GPOs) in order to target the software to specific users and groups. In addition, you begin to create a pilot program to test your deployment before actually going live with your rollout. (We discussed creating and modifying GPOs in Chapters 10 and 11; you can refer back if you have any questions about using GPOs to target settings at specific users, computers, and groups.)

The targeting phase involves extensive use of the Software Installation snap-in, which is discussed in detail later in this chapter. So, we'll hold off on a more involved discussion of targeting software.

Pilot Phase

Of all the phases involved in software management, the pilot phase most often gets shortchanged in small to medium-sized organizations. A pilot program is a trial run deployed first in a lab environment and then to a subset of users, for the express purpose of troubleshooting and debugging any application issues prior to the full-blown deployment. A good pilot program will also provide the foundation for the ongoing maintenance of the software, from being able to apply patches or upgrades to being able to remove the application.

The pilot phase is often skimped on by smaller companies due to resource availability. Often, small to medium-sized companies cannot afford or choose not to carry extra inventory that can be used as a research and development (R&D) or test lab. Perfectly good

computer systems that are not being used for productive means are a poor investment, in the minds of many small companies. What they do not realize, however, is that having a lab for testing new software, whether full applications or just upgrades or patches, actually carries the cost benefit of reducing downtime associated with rolling out applications into a production environment and troubleshooting live systems on the fly as users wait impatiently. Successfully piloting an application first strongly increases the likelihood of a successful deployment. If you work for a company that is hesitant to devote resources for pilot programs, ask management if they would consider receiving medical treatment if someone developed a drug and offered it directly to the public with no prior testing (disregard FDA regulations for a moment). Chances are, they will say absolutely not. So, why risk the health of their computer networks, on which their business relies, to software that has not been tested in your environment?

Elements of a Pilot Program

A pilot program begins in the lab, on non-production machines. Then it does not matter if the software crashes the computer and forces you to have to reinstall the operating system. Once the software is proven to work in your lab environment, you are ready to deploy it to a subset of the users on your network. Why deploy to only a subset of users and not everyone? Well, one thing every good administrator learns (often from experience) is that application behavior in a lab environment, no matter how closely the lab simulates the production environment, is often different than in the production environment. When you deploy to live users, you will invariably discover strange new issues with the software. It is much easier to troubleshoot and fix a software deployment to 20 users than it is to 2,000.

User Types

When considering the target group for the pilot, it is important to select a group that will be representative of the organization as a whole. They should run all the regular applications in use by the company and be the very definition of the average users. If you have mobile users who work from both home and the office, you should include someone from that group in the pilot and test the software under both conditions. Assemble your pilot group so that all the types of users you have in the organization are part of the process.

How to Test

A good pilot should test every possible manner in which you will deploy the software application. Will you be publishing the application to some users and assigning the application to other users? Will you be adding any custom files to the installation, such as templates? All of these elements need to be tested to ensure that they function correctly before the rollout.

In addition, the pilot is the opportunity to carry out performance testing on the software. If the application is installed locally to user hard drives, does it perform adequately for the average computer systems used by the company? Will some systems need upgrades to run

the application more efficiently? If it is a network application, can the network handle the stress of multiple users running the application at once? Will you need to add server capacity, or possibly replace hubs with switches or segment your network with routers in order to make more efficient use of bandwidth? For obvious reasons, it is much better to answer these questions during the pilot program than to make a guess and spend money unnecessarily for upgrades you don't need. Even worse would be overestimating your capacity and watching production grind to a halt when users can't access the necessary resources, or when the applications perform so slowly that they might as well be unusable.

Installation Phase

Once you are satisfied that you've answered all the questions in the pilot program and you are confident of the application's ability to function as expected in your environment, you are ready to proceed to the installation phase. In this phase, you roll out the software across your organization to all users and computers who are set up to receive it.

The installation phase is invoked whenever you need to install an application fresh, update an existing application, or patch an existing application. For Windows 2000, usually you will use the Windows Installer to install software and manage an installation, unless you are dealing with third-party, non-Windows 2000-certified software. The Windows 2000 CD comes with the VERITAS WinINSTALL Discover utility, which can be installed separately. (Seagate owned WinINSTALL Discover before VERITAS bought it.) In some Microsoft (and non-Microsoft) references and documentation, you will see WinINSTALL Discover referred to as WinINSTALL LE, although in this chapter we will refer to it simply as WinINSTALL.

WinINSTALL enables you, as the administrator, to create or modify Windows Installer packages and to repackage third-party software as Windows Installer packages. There are some limitations to this process, however, as we will see in the next section.

Creating and Configuring Windows Installer Packages

Later in this chapter, you will learn how to deploy software on your Windows 2000 network, but for now let's take a look at actually creating some software packages to deploy. Windows 2000 includes the Windows Installer file format as its default software installation packaging, a format first introduced with the Microsoft Office 2000 suite. In this section, we will discuss the following aspects of the Windows Installer:

- Windows Installer technology overview
- Managing Windows Installer settings with Group Policy
- Creating Windows Installer packages with WinINSTALL LE

Windows Installer Technology Overview

As we've briefly mentioned already, Windows Installer is a client-side technology that allows for a more intelligent software deployment than has been available in the past. Windows Installer technology is built in to Windows 2000; unlike many of the new technologies, however, it can also be used in Windows 9*x* and NT 4. Office 2000 was the first released product to incorporate the Windows Installer technology, and other applications have followed. In fact, in order for an application to be certified as Windows 2000 compatible, it must use the Windows Installer.

When a Windows Installer package is installed for the first time on a Windows 9*x* or NT 4 system, Windows Installer technology is installed and configured on the computer in order to support the software installation. Microsoft provides a freely redistributable installation program called instmsi.exe to enable Windows Installer support on these platforms. Because Windows Installer is already on Windows 2000 systems, no updates are needed prior to the application installation.

We say that Windows Installer allows for a more intelligent software deployment than has been available in the past because of the new features it provides. They are:

- *Software diagnostics and repair*—Windows Installer makes the concept of self-healing applications a reality. If a user accidentally deletes a critical application DLL file, for example, Windows Installer will automatically recognize this action and replace the missing file from the installation source files. In the past, this type of error would have been a show stopper, and almost invariably it would have resulted in a support call and a trip from a desktop support technician to reinstall the application.
- *Complete uninstallation of software*—Often in the past, uninstalling a program from your system resulted in numerous files, directories, and Registry settings being left behind by the removal program. These orphaned files were, at minimum, an annoyance that wasted disk space and memory resources. In some cases, orphaned DLL files could interact with other programs and cause problems that were difficult to diagnose. Sometimes, the uninstall process would remove shared files that were used by other programs, causing those programs to no longer function correctly. Windows Installer manages the entire installation process, so it is able to remove a program completely should you choose to uninstall it. Windows Installer also prevents shared files from being removed, because it knows which other Windows Installer applications might be using those files.
- *Multiplatform capability*—Unlike traditional installation programs that package software for a particular platform (such as Windows, MacOS, or Solaris), Windows Installer technology enables developers to create a single installation package that can install the software on different platforms and in different configurations. This ability reduces the amount of overhead associated with packaging an application for distribution.

Now you know what the Windows Installer is capable of. Let's see what a Windows Installer package is.

Windows Installer Packages

A Windows Installer package is an MSI file that contains instructions pertaining to the installation and removal of an application. The package comes in the format of a relational database, and it contains all the information necessary to manage the program's installation.

In addition to MSI files, a Windows Installer routine can also include MST files, which are known as **transforms**. A transform is used to customize the installation of an application, changing it from its default behavior. This file is put in the same installation directory as the MSI file, and it can be modified at any time.

Windows Installer uses a transaction-based system similar to that used in Active Directory and Microsoft Exchange Server. A transaction-based system writes changes one at a time, so that at any point of failure, the installation can be rolled back to its prior state. This is a great feature for administrators, who have long suffered the pain of having an application's installation program crash and leave temporary files and a half-installed application on the hard drive and in the Registry with no easy means of removing them. With Windows Installer, a failed installation will simply roll back to its state prior to the installation's beginning. You will learn how to package programs with third-party Windows Installer packager later in this chapter.

Windows Installer settings can be managed with the Group Policy Editor. We'll look at this process next.

Managing Windows Installer Settings with Group Policy

Windows Installer has global settings in Windows 2000 that affect the way it behaves when Windows Installer–compatible software is installed. These settings are managed on a GPO basis through the Group Policy Editor. As you will remember from the last couple of chapters, GPOs are assigned to computers, users, and groups. If you want to have Windows Installer settings that act globally in your Active Directory, edit the default domain policy GPO when configuring the settings.

As we've discussed previously, in many cases Group Policy settings can be applied either to computers (the Computer Configuration container) or to users (User Configuration). That is the case with Windows Installer, which uses different settings depending on whether you are configuring the policies for computers or users. Figures 12-1 and 12-2 illustrate these differences.

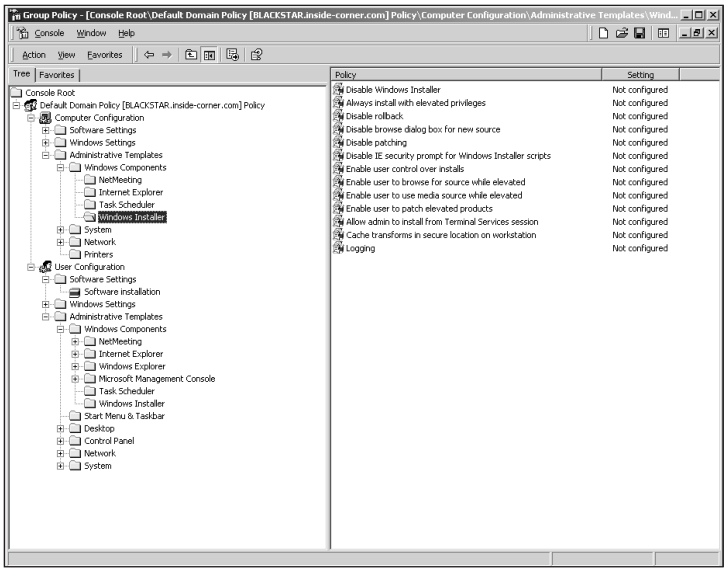

Figure 12-1 The Windows Installer settings for computers

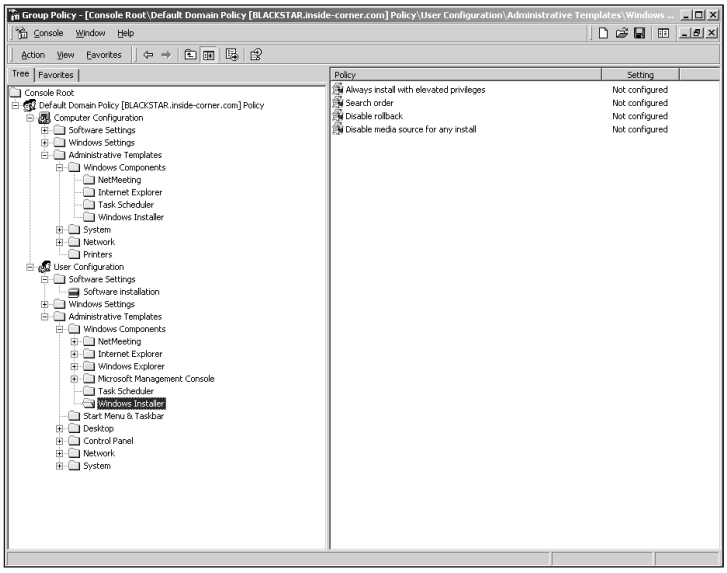

Figure 12-2 The Windows Installer settings for users

Some of the key settings include:

- *Always Install With Elevated Privileges*—Available under both the Computer and User Configuration containers; allows Windows Installer packages to be installed with administrator privileges. These privileges are usually required in order to install software on a Windows 2000 system, because an ordinary user account cannot make changes to the Registry.
- *Disable Media Source For Any Install*—When enabled, prevents ordinary user accounts from being able to install software from removable media drives (CD or floppy disk, typically).
- *Enable User To Use Media Source While Elevated*—Enables or disables a user's ability to install software from removable media under elevated permissions. This setting is used in conjunction with Disable Media Source For Any Install to prevent users from installing their own software from CD or floppy disk. The purpose is to keep users from loading unauthorized software on their systems (a common problem in many companies).
- *Disable Windows Installer*—Restricts a user's ability to install Windows Installer–packaged software. There are three settings: Never, which fully enables Windows Installer (the default setting); For Non-Managed Apps Only, which allows users to install only applications assigned or published by the administrator; and Always, which prevents users from installing any Windows Installer–based application at all.

Typically, you won't need to restrict Windows Installer settings unless you are working in a high-security environment. The only setting that we recommend configuring for the average environment is Always Install With Elevated Privileges. For most organizations, this convenience saves many support calls while posing a minimal amount of security risk.

In addition to managing Windows Installer settings, you can also create your own Windows Installer packages.

Creating Windows Installer Packages with Win INSTALL LE

As we discussed earlier in this chapter, Windows 2000 Server offers the third-party program WinINSTALL LE from VERITAS Software. This program can be installed from the \VALUEADD\3RDPARTY\MGMT\WINSTLE folder on the installation CD. Two components are installed: VERITAS Discover (WinINSTALL Discover) and VERITAS Software Console. The Software Console contains the Windows Installer Package Editor, which we will discuss later in this chapter. In this section, we will focus on WinINSTALL Discover, which will allow you to create a new Windows Installer package. In our example, we will create a package for WinZip 8.0. (WinZip 8.0 is available from WinZip Computing, Inc. WinZip is a registered trademark of WinZip Computing, Inc. [copyright 1991–2000], and is available from www.winzip.com. WinZip screen images are reproduced with permission of WinZip Computing, Inc.)

Before creating a package, you must keep in mind an important consideration. As a rule, you should use a clean operating system to create a package—that is, one that does not already have a bunch of other applications installed. The reason is that usually, a software installation program will not copy a file to a hard drive if a newer version exists. If you create a package on a system that has a particular application other systems do not have, and you then deploy the package to other systems, you might find that your package does not work on those other systems due to missing files. Also, for the package-creation process, you should use a system that is representative of the average computer on your network. The more like other computers your development system is, the more likely you are to end up with a good Windows Installer package that works flawlessly on other systems.

Creating a new Windows Installer package—an MSI file—first involves launching the WinINSTALL Discover application from the Start menu. Figure 12-3 shows the screen you will first see.

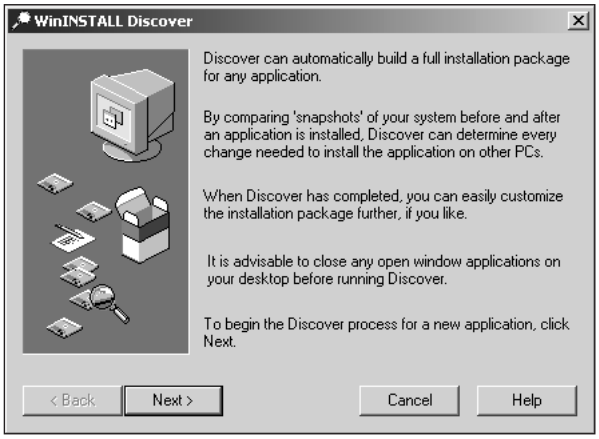

Figure 12-3 The WinINSTALL Discover welcome describes the Windows Installer package creation process

When you click on Next, you will be prompted to supply the filename for your new package. As a reminder, Windows Installer packages have an .msi extension. Supply a descriptive filename for your package and choose the directory the file will be saved in, as in Figure 12-4.

On the next screen, shown in Figure 12-5, you will find that the path and filename for the new package are already filled in from your choices on the previous screen. You can modify those choices now, if you want. In addition, you are prompted to name the application. This name will become the display title in the Add/Remove Programs applet. You can also change the language setting if necessary. Click on Next.

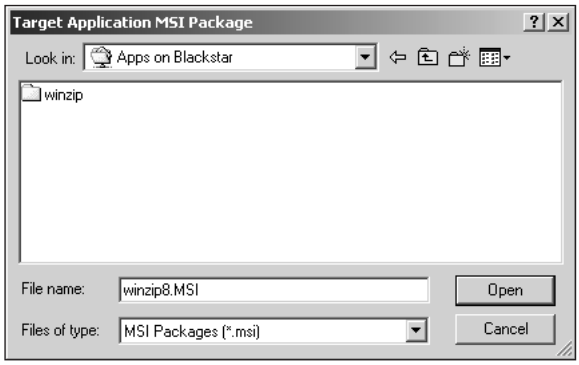

Figure 12-4 The first step in creating a package is to name the MSI file and choose an appropriate directory in which to store the new package

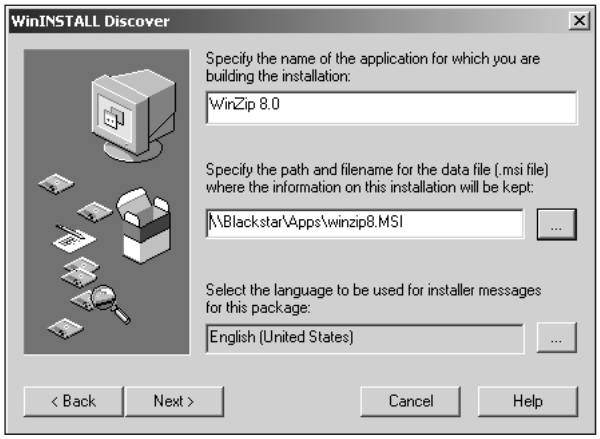

Figure 12-5 The next step in creating a new package is to name your application and verify that the path and filename are correct

After naming your application, you are prompted for the drive that WinINSTALL Discover will use to store temporary files during the package creation process. You should choose a drive that contains plenty of free disk space, depending on the application. Figure 12-6 illustrates this dialog box.

The next step is to determine what drives WinINSTALL should check for changes, as shown in Figure 12-7. In our example, we have chosen to have WinINSTALL scan drives C: and E:, because we intend to install our application to C:, and E: is the drive where our Windows 2000 operating system is installed. D: is strictly a data drive in our configuration, so we have chosen to skip scanning it for changes. In most cases, you will probably want WinINSTALL to scan all local hard drives just to make sure no changes slip by.

Figure 12-6 You must choose a drive to store temporary working files for WinINSTALL Discover

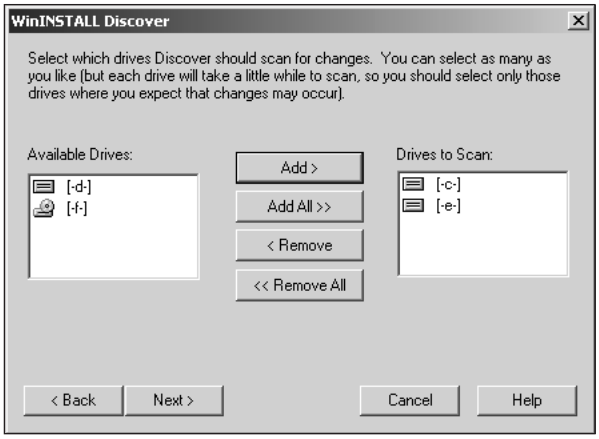

Figure 12-7 WinINSTALL Discover will scan the drives you choose for changes made during the application installation

WinINSTALL then asks if you want to exclude any files or folders from the scan, as illustrated in Figure 12-8. You should choose to exclude obvious files like pagefile.sys, because they could change during the installation and you wouldn't want them included as part of your distribution. Another likely candidate to exclude would be the Recycle Bin. Make your choices and click on Next.

Next, WinINSTALL will begin to scan with the criteria you specified, as shown in Figure 12-9. Once the scan is complete, you will receive the dialog box shown in Figure 12-10. The initial "before" scan will look at every file on your hard drives (those that you specified for scanning) and every Registry setting. After the application is installed, an "after" scan will run on the same drives and Registry and compare the before and after states. The changes that occurred between the two scans are compiled into the Windows Installer package.

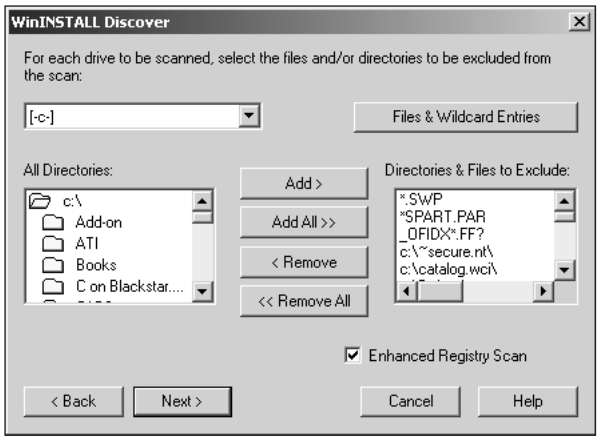

Figure 12-8 WinINSTALL allows you to exclude files and folders that could possibly change during installation but that should not be included with the distribution

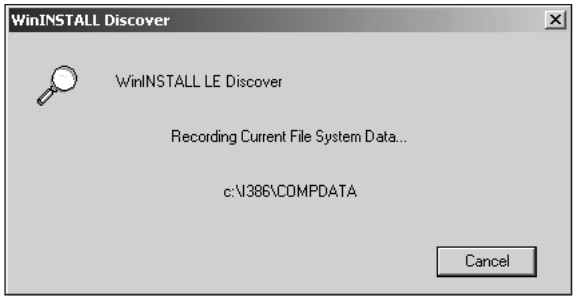

Figure 12-9 WinINSTALL takes a snapshot of the current state of your system before running the application installation

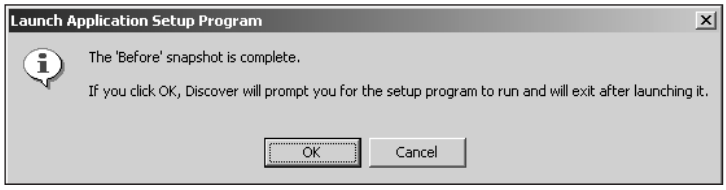

Figure 12-10 Once the initial scan is complete, WinINSTALL will begin the application installation process

After you click on OK in the dialog box in Figure 12-10, you are prompted to choose your application's installation program. Figure 12-11 shows this dialog box.

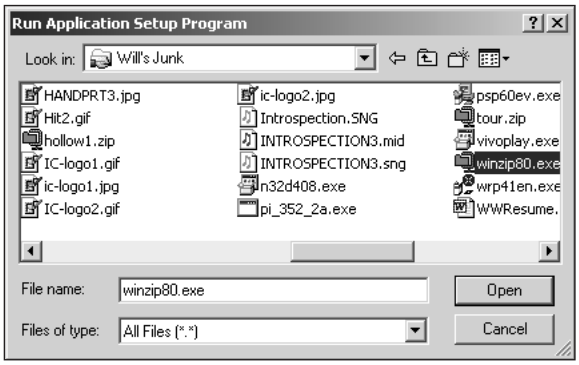

Figure 12-11 To begin installation of your application, simply navigate to the source directory and select its setup program

The application installation proceeds at this point, as it would if you were installing the application normally and not building a Windows Installer package. Install the program as you want it to be configured for your package, with any desired options. Once you have installed the program, *do not* reboot if you are prompted to. If you reboot, Registry changes or file updates might be committed that would not be possible later when you are installing the package on another machine, causing that installation to fail. If the program requires a reboot, it has Registry entries that execute and complete the updates upon reboot. You want future machines receiving this package to be able to make these updates as well, to ensure a successful installation.

Rather than reboot, choose to return to Windows or to restart later, whatever the wording might be. WinINSTALL has exited at this point, so relaunch it through the Start menu. You will then see the dialog box shown in Figure 12-12, which gives you the option of discarding your package or continuing with the "after" snapshot.

As shown in Figure 12-13, WinINSTALL repeats the scanning process to determine the differences between pre-installation and post-installation. Once the snapshot is complete, the dialog box in Figure 12-14 appears, advising you that your Windows Installer package has been successfully created.

With your package created, you are ready to use the Software Installation Group Policy snap-in to manage and deploy it.

Figure 12-12 You can now continue your package creation by performing the "after" snapshot on your system

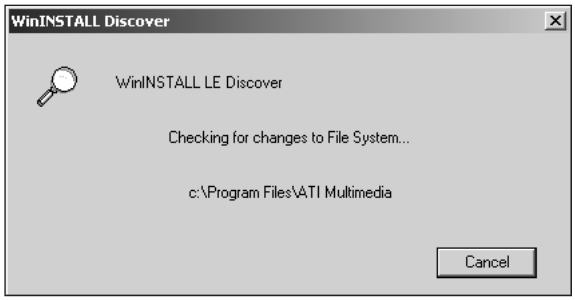

Figure 12-13 WinINSTALL repeats the earlier scan, checking for differences

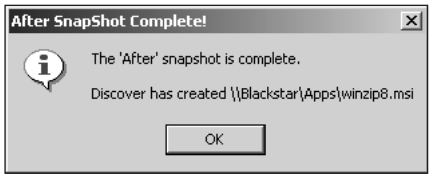

Figure 12-14 WinINSTALL notifies you that your Windows Installer package has been successfully created

Managing Software Deployment through the Software Installation Snap-in

The Software Installation snap-in is an extension to Group Policy that allows you to establish a Group Policy–based software management system. As a component of IntelliMirror, Software Installation works to reduce TCO for your network and increases the efficiency of both end users and IT staff members. Through Software Installation, you can centrally manage each of the following:

- *Deploying applications*—These can be shrink-wrapped third-party software or custom-built applications developed in-house.
- *Applying upgrades and patches*—Through Software Installation, you can update existing software or even replace it in the case of a product upgrade. Deploying service packs for operating systems becomes much easier, as well.
- *Uninstalling software*—When a product is no longer in use or supported by the IT department, you can easily remove it from users' computers.

The goal of Software Installation is to deploy applications in such a way that whenever users log on to a computer, no matter where they are, their applications will always be available to them. When combined with the Windows Installer, this technology is often referred to as just-in-time (JIT), because deployment will occur either during user logon or when the user launches a particular application. For example, say you have assigned Microsoft Word to a particular user. Even though users have not explicitly installed Microsoft Word, they see the icon for it on their desktop or in their Start menu. The first time they attempt to use Word, the system installs the application automatically, with no user intervention, and launches the program.

Likewise, if users attempt to use a feature of the program that is not installed by default, the application will be smart enough to automatically install the missing feature on the fly from the network and allow users to use it. In the past, installing a missing feature invariably meant manually running the program's setup utility and either reinstalling the entire product to add the missing feature or simply selecting the missing feature and choosing to update the installation. In either case, it interrupted the user's workflow and very likely required a desk-side trip from a desktop support technician.

We will discuss assigning applications through Software Installation a bit later.

Requirements for Software Installation

In order to use the Software Installation snap-in, several prerequisites must first be met. In addition, some conditions apply, depending on what type of application you are attempting to manage. The following sections discuss these Software Installation topics.

Group Policy Dependency

In order to use the Software Installation snap-in, you must be using Group Policy on your network. Because Group Policy is limited to Windows 2000 computers, you will be able to manage software only for your Windows 2000 environment. Any legacy Windows 9*x* or NT 4 clients will not be able to receive applications through Group Policy and Software Installation.

Active Directory Dependency

Because Group Policy is dependent on Active Directory, it makes sense that you cannot use the Software Installation snap-in unless you have deployed Active Directory on your network. Software Installation relies on GPOs, which are stored in Active Directory, to determine who can access software that it manages. As with Group Policy, Windows 9*x* and NT 4 computers cannot participate in Active Directory. At this point, Microsoft has not determined whether it will include Active Directory support in its upcoming Windows Millennium operating system. Windows Millennium (sometimes written Windows ME or WinME) is the next-in-line consumer operating system that will replace Windows 98. Because Microsoft is aiming Windows Millennium at the consumer market and Windows 2000 at the business market, it has not decided for certain whether it will force all businesses using the Win9*x* line of operating systems to upgrade to the more expensive and hardware-intensive Windows 2000 in order to take advantage of the features of Active Directory.

Works Best with Windows Installer Applications

By combining Software Installation with Windows Installer–based applications, you can make JIT deployment of applications a reality. Rather than producing an error message or just crashing, applications can be self-healing and automatically replace missing features or files when they are accessed. Applications that are Windows 2000 certified will have been created as Windows Installer packages.

Works with Non–Windows Installer Applications

Although limitations exist, Software Installation can be used to deploy non–Windows Installer packaged software, as well. The main limitation is the lack of support for JIT delivery of application settings and features. Existing setup programs should be repackaged as Windows Installer packages, if for no other reason than the fact that installing software usually requires administrative privileges on the local system. Through Group Policy, you can have Windows Installer packages always install with elevated privileges, meaning ordinary users can install software without administrator intervention.

You can also deploy non–Windows Installer packages by creating a ZAP file, which is sort of like an old INI file. It contains settings relevant to controlling the program's appearance and behavior. You create the ZAP file in a text editor and place it in the shared folder with the application's setup program. Here is a sample ZAP file for deploying Microsoft Office 97:

```
[Application]
FriendlyName = "Microsoft Office 97"
SetupCommand = ""\\Server01\apps\Off97\setup.exe""
[ext]
DOC=
DOT=
RTF=
XLS=
XLA=
XLW=
MDB=
PPT=
```

This example file contains two sections: [Application] and [ext]. The [Application] section is the only required section within a ZAP file, and the entries shown in our example are the only required elements. FriendlyName is the descriptive name that is displayed in the Add/Remove Programs applet, and SetupCommand provides the universal naming convention (UNC) path to the application's setup file.

The second section, [ext], is strictly optional. It lists file extensions for the program that should auto-install into the Registry.

Other optional sections can be defined within a ZAP file, but they are outside the scope of this book. The Windows 2000 Resource Kit has an excellent section on creating ZAP files for software distribution.

Assigned versus Published Applications

As we discussed earlier, applications can either be assigned or published to users. Assigned applications appear in the Start menu or as desktop icons, and reappear even if users uninstall the applications through the Add/Remove Programs applet in Control Panel. Published applications are made available through Add/Remove Programs, but a user must explicitly install them in order to use them. Unlike assigned applications, published applications do not stick with the system if a user uninstalls them.

We'll discuss the method of assigning and publishing applications shortly.

Using the Software Installation Snap-in

To this point you've learned a lot about what the Software Installation snap-in is and does, but nothing about how to actually use it. In this section, we will explore the Software Installation snap-in, specifically the following topics:

- Configuring Software Installation properties
- Deploying a new package
- Configuring package properties

Configuring Software Installation Properties

Before we get into deploying packages to manage with the Software Installation snap-in, let's first look at the global settings that can be configured. The Software Installation snap-in is located under both the Computer Configuration and User Configuration containers of a GPO. Open the GPO to which you wish to assign a software package; as you can see in Figure 12-15, the Software Installation snap-in is located under the Software Settings node.

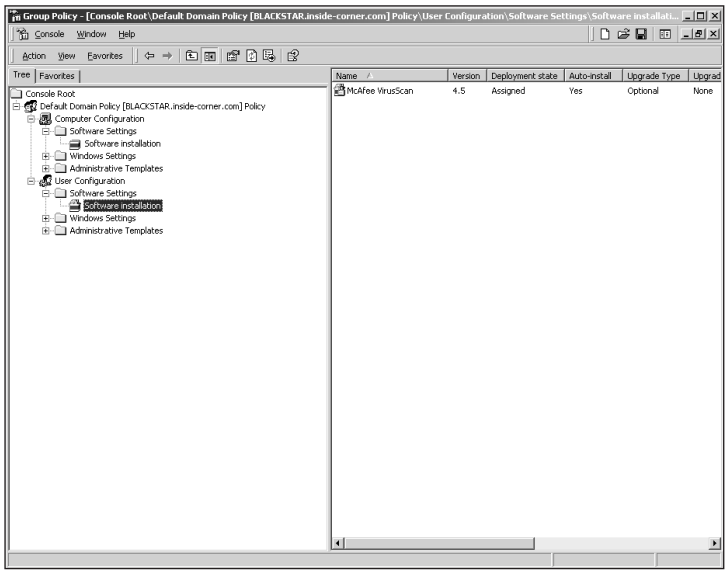

Figure 12-15 The Software Installation snap-in is located under the Software Settings node in both the Computer Configuration and User Configuration containers of a Group Policy Object

To configure the global properties of the Software Installation snap-in, simply right-click on the snap-in and choose Properties. Keep in mind that computer and user settings are independent of each other, so making changes to the computer policy for Software Installation will have no effect on the user policy, and vice versa.

The first dialog box you are presented with when you enter Software Installation Properties is the General tab, shown in Figure 12-16.

The first section allows you to define the default package location for new packages. This setting is useful if you collect all your packages to be deployed into a centralized location.

The next section defines the behavior of the snap-in with regard to new package creation. By default, the Deploy Software dialog box will be displayed when you choose to create a new package. This dialog box contains the choice to assign or publish a package, allowing you to choose how you want Software Installation to handle a package on a package-by-package basis.

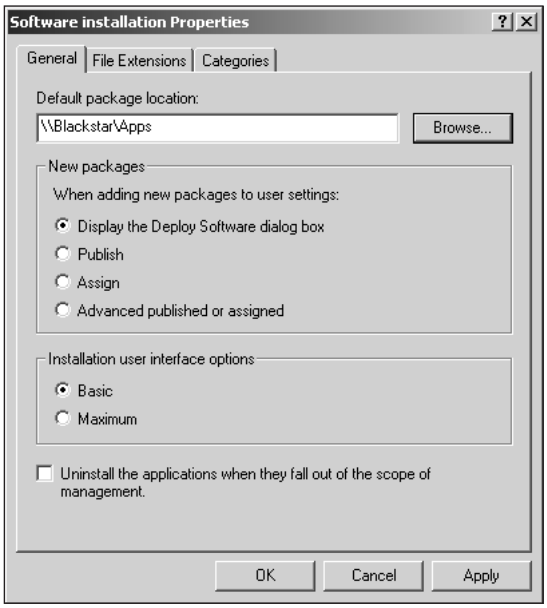

Figure 12-16 The General tab contains information about the default behavior of the Software Installation snap-in

In addition, the General tab contains a setting to determine how much information is presented to the user during package installation. By default, only basic information and options are supplied. Optionally, you can specify that a maximum amount of information and options be shown to the user during installation.

Last, the General tab has the optional setting to define whether software should automatically be uninstalled when it falls outside the scope of management. That is, if Software Installation no longer manages applications, they should no longer be available to users. By default, this option is turned off.

After the General tab is the File Extensions tab, as shown in Figure 12-17. Often, more than one application installed on your computer is capable of opening a given type of file. This section allows you to pick a file type and set the order of precedence for applications that are capable of opening the application.

The last tab is Categories, which is an organizational option. You create categories to help keep track of where software is deployed. By default, there are no categories; you must create them if you want to use this feature. You name your categories according to personal preference, although your environment will probably dictate the names. For example, you might choose to create categories for your software based on department or location. Figure 12-18 shows an example of the Categories property sheet.

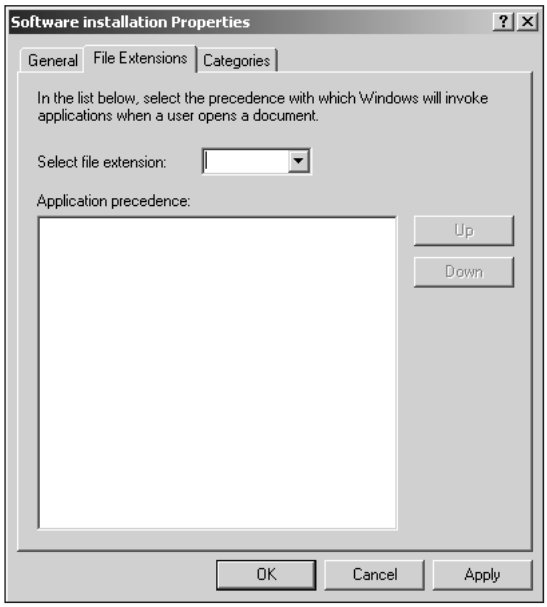

Figure 12-17 The File Extensions tab allows an administrator to set a precedence for programs that are capable of opening a given file type

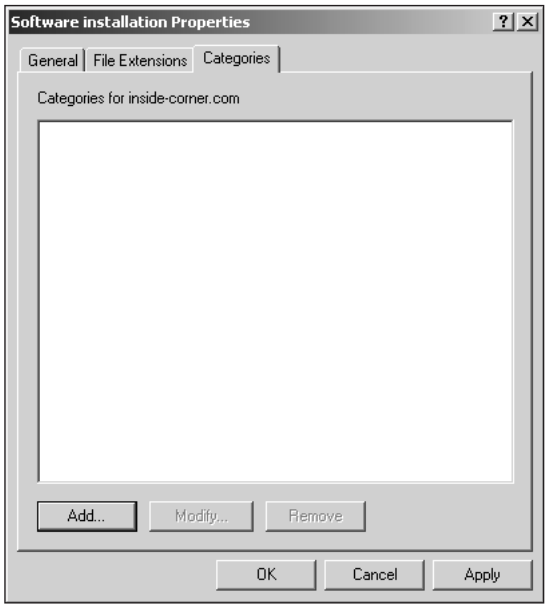

Figure 12-18 The Categories tab allows you to organize your managed software into custom-defined categories

Deploying a New Package

With the global options set for Software Installation, we will move on to deploying a new package. In order to deploy a new package, you must have first copied the installation files to a network share, otherwise known as a distribution point. Right-click on the Software Installation snap-in and choose New|Package. The dialog box shown in Figure 12-19 will appear.

Figure 12-19 The first step in deploying a new package is to select the Windows Installer package or ZAP file that is to be deployed

In our example, we're selecting a Windows Installer package for McAfee VirusScan 4.5, which is located in the VirusScan share on the server Blackstar. This is our distribution point. When we select the file and click on Open, we are presented with the dialog box shown in Figure 12-20. We see this dialog box because previously we left the global setting for Software Installation to show us these choices when creating a new package, rather than to default to either publishing or assigning.

It is important to note that you can publish an application only if the package is being deployed under the User Configuration container. Software deployed to computers does not support publishing, and therefore those packages can only be assigned. If you deploy a package under the Computer Configuration container, when you reach the dialog box shown in Figure 12-20 the Published option will be unavailable.

If you select either Published or Assigned and click on OK, the package is deployed without any further prompting. If you select Advanced Published Or Assigned, the package will still be deployed, but you will be prompted with a dialog box similar to that shown in Figure 12-21. This is the same dialog box you can access later by going into the properties of a package, as discussed next.

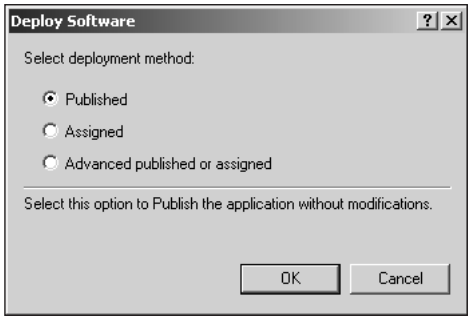

Figure 12-20 After choosing the software package to deploy, you must decide whether to publish or assign it

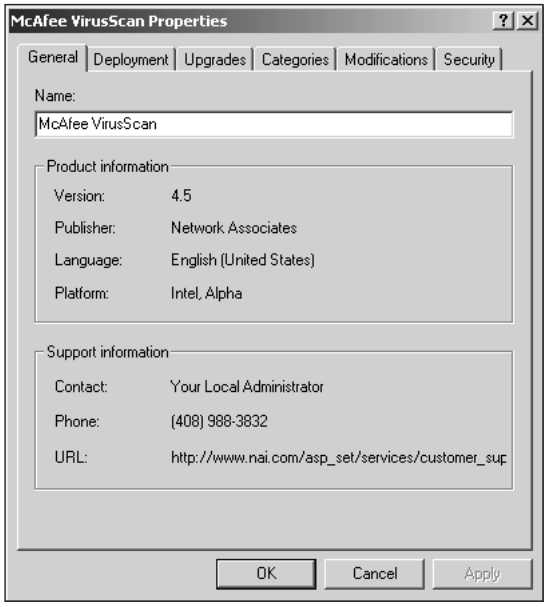

Figure 12-21 You can configure a number of advanced settings for an application once it has been deployed

Configuring Package Properties

To access the properties of a package once you've deployed it, simply right-click on the package and choose Properties. You will see the same dialog box you saw if you selected

Advanced Published Or Assigned during the new package deployment. A number of property sheets contain settings for the package. An overview of them is as follows:

- *General*—Contains product information such as the name and version number, as well as contact information.
- *Deployment*—Defines the deployment type (assigned or published), which can also be changed here. In addition, this property sheet contains settings for deployment options, including whether the package should be uninstalled if it falls outside the scope of management and if it should be displayed in the Add/Remove Programs applet. Advanced deployment options determine whether the language setting should be ignored when installing the software and whether previous installations of the product should be uninstalled if they weren't installed through Group Policy.
- *Upgrades*—Defines the applications that are to be upgraded by this package, and which packages can upgrade this package.
- *Categories*—Determines the categories under which the software will be displayed in the Add/Remove Programs applet.
- *Modifications*—Allows you to apply modifications or transforms to the package in order to customize the deployment.
- *Security*—Determines who has what level of access to the package. Through the Security property sheet, you control the deployment of the software to computers, users, and groups.

These are the basics of the Software Installation snap-in. With an understanding of the Windows Installer and Software Installation, you are ready to put both of them together to deploy software on your Windows 2000 network. However, sometimes an existing package will no longer work in a changing environment. A package may simply have to be re-created from scratch. In many cases, however, it is easier to patch an existing package.

Patching Software Using the Windows Installer Package Editor

After you've created a Windows Installer software package, a time may come when you need to modify it. Patching software is the process by which you modify a Windows Installer package to meet changing needs. You perform this task through the VERITAS Software Console utility (also called the Windows Installer Package Editor) that is installed as part of WinINSTALL LE. Figure 12-22 shows the WinZip 8 package that we just created opened in the Software Console for modification.

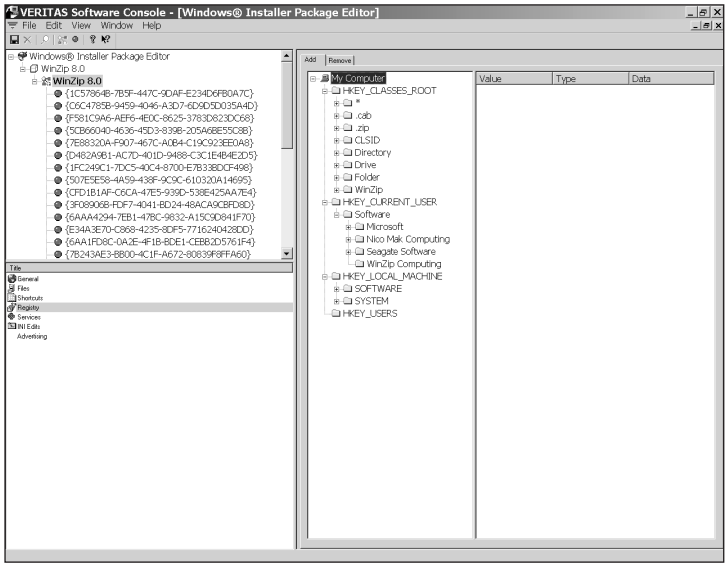

Figure 12-22 The VERITAS Software Console enables you to open and modify existing Windows Installer packages

You can see that Software Console is divided into three main sections. They are:

- *Tree view (top-left frame)*—Shows the Windows Installer Package Editor header with a listing of the current package (if any) open for editing.
- *List view (bottom-left frame)*—Contains the heart of the package information. The information about the package in this section includes:
  - *General*—Includes a summary of installed features.
  - *Files*—Shows a list of the files that will install with this package, as well as their size, version number, and target locations.
  - *Shortcuts*—Shows any shortcuts that will be created by the package during installation.
  - *Registry*—Shows a list of Registry keys and values that the installation will update. The example in Figure 12-22 shows the Registry updates for our WinZip package.
  - *Services*—Shows a list of new services that will be installed on Windows NT and Windows 2000–based computers.
  - *INI Edits*—Exists primarily for backward compatibility with 16-bit applications. This setting shows any INI files that will be updated during the installation, as well as the specific modifications to be made.

- *Advertising*—Contains COM settings for interaction with a COM server, and includes file types that will be registered by the package installation.

- *Data view (right frame)*—Shows the contents of the selections made in the list view. For example, if you select Services in the list view, the data view will show the services to be installed.

Patching Applications

Patching (modifying) a Windows Installer package is not too difficult, once you are familiar with the layout of the Software Console as described in the previous section. At the top of the data view frame is a tab for an Add property sheet for most list view settings, and a Remove tab for a few, as well. On the Add tab, for instance, are New, Properties, and Delete choices. These choices allow you to manipulate the contents in the data view, such as adding custom files to your installation that are not included by default or removing settings and features you no longer need.

In some cases, you might use the Windows Installer Package Editor to clean up a package before distributing it. Say, for example, you forgot to exclude the page file on your Windows 2000 system from the "before" and "after" scans, and it inadvertently ended up in your distribution file. Using the Software Console, you could remove the file from the package so it doesn't needlessly increase the file size of the package and potentially cause installation problems when it can't overwrite an existing page file during installation. You can also customize the package for different deployments if you wish, saving the modifications with different file names.

If you patch a package that has already been deployed to other users, you can use the Software Installation snap-in in Group Policy to automatically patch/upgrade the existing application on users' systems, as discussed previously in this chapter.

Chapter Summary

In this chapter, you learned about the powerful software deployment and management features of Windows 2000, which are part of Microsoft's IntelliMirror technologies. Some of the key topics included the following:

- Change and configuration management is an increasingly important IT concern. The soft costs of supporting a network far exceed the hard costs of purchasing the actual equipment.

- IntelliMirror consists of data management, desktop settings management, and software installation and maintenance.

- IntelliMirror technologies are dependent on Active Directory and Group Policy in order to operate.

- The phases of software management can be broken down into preparation, distribution, targeting, pilot, and installation.
- Windows 2000 includes VERITAS WinINSTALL LE on the installation CD for creating and patching Windows Installer packages.
- Windows Installer packages end with an .msi extension.
- Windows Installer packages can be self-healing, automatically detecting missing files and settings and replacing them on the fly when combined with the Software Installation snap-in.
- The Software Installation utility requires Group Policy.
- Software can either be assigned or published to users through Software Installation.
- Windows Installer packages can be patched after they've been created, to remove incorrect files and settings, add additional customizations, and change the installation behavior.